2014 IEEE 11th International Conference on e-Business Engineering

Optimistic Fair-Exchange with Anonymity for Bitcoin Users

Danushka Jayasinghe, Konstantinos Markantonakis and Keith Mayes
Smart Card Centre, Information Security Group,
Royal Holloway, University of London, Egham, Surrey, UK, TW20 0EX
Email: {Danushka.Jayasinghe.2012, K.Markantonakis, Keith.Mayes}@rhul.ac.uk

***Abstract*—Fair-exchange and anonymity are two important attributes in e-commerce. It is much more difficult to expect fairness in e-commerce transactions using Bitcoin due to anonymity and transaction irreversibility. Genuine consumers and merchants who would like to make and receive payments using Bitcoin may be reluctant to do so due to this uncertainty. The proposed protocol guarantees strong-fairness while preserving anonymity of the consumer and the merchant, using Bitcoin as a payment method which addresses the aforementioned concern. The involvement of the trusted third party (TTP) is kept to a minimum, which makes the protocol optimistic and the exchanged product is not revealed to TTP. It achieves dispute resolution within the protocol run without any intervention of an external judge. Finally we show how the protocol can be easily adapted to use other digital cash systems designed using public ledgers such as Zerocoin/Zerocash.**

***Keywords*—Anonymity, Optimistic Fair-Exchange, Bitcoin.**

I. Introduction

The e-commerce market is growing and is expected to double from a value of €755 billion in 2010 to an expected €1460 billion in 2015 [19]. Yet there are issues with regards to the anonymity and fairness of transactions in an e-commerce environment. In a traditional Point-of-Sale (POS) transaction, a consumer can maintain anonymity by using cash rather than handing personal financial details to the merchant and the correctness of the goods purchased can be immediately verified. In an E-commerce scenario, however, where transacting parties do not see each other physically, it is much easier for a dishonest party to misbehave and payment is not usually anonymous. Anonymity prevents merchants and other parties from learning consumer personal information, spending habits and financial details. This would also help reduce fraudulent activity related to theft of consumer financial details. Currently the majority of e-commerce does not fully offer anonymity, although services such as PayPal can be used to hide personal financial details from the merchants.

As a result there have been attempts to give e-commerce users the freedom of making anonymous payments without having to reveal personal financial details and to guarantee fair-exchange [21], [22], [20], [17]. The main contribution of this paper is the optimistic fair-exchange protocol proposed in section IV. In recent years there has been a dramatic increase in the use of Bitcoin as an alternative payment method in e-commerce. Due to the fact that Bitcoin payments being anonymous and irreversible, it is much more difficult to expect fairness in e-commerce transactions while using Bitcoin. Genuine consumers and merchants who would like to make and receive payments using Bitcoin may be reluctant to do so due to this uncertainty. There are current solutions such as "e-bay Guarantee" and "PayPal Buyer Protection" for non-Bitcoin transactions. Even though, these methods have capped transaction values or involve lengthy dispute resolutions, they provide a level of peace of mind for consumers. Currently there seems to be no option for Bitcoin users to enjoy such peace of mind in e-commerce. Bitcoin has now become a competitive player for alternative payment methods with a considerable market capitalisation. yet, to our knowledge there are no specifically designed e-commerce protocols for Bitcoin that addresses aforementioned concerns and this paper is one of the first Bitcoin-specific proposals that guarantees strong-fairness and anonymity.

The rest of the paper is structured as follows. In Section II, we explore related anonymous payment protocols. Section III examines related fair-exchange protocols. In Section IV, we propose our protocol and in section V, a security analysis of the protocol is carried out. In Section VI, we add an extension to our protocol to support the improved Zerocash system for Zerocoin. Finally in Section VII, we conclude our discussion and provide further research directions.

II. Anonymous Payment Protocol Schemes

Protocols that help realise anonymity and user privacy during payment are called *Anonymous Payment Protocols.* Anonymous payment protocol systems can be categorised as on-line payment systems and off-line payment systems. In an *on-line scheme* the payer, payee and the bank needs to be connected on-line at least once during the protocol for verification of coins. Proposals include [10], [8], [9]. In *off-line schemes* the Trusted Third Party (TTP) does not have to be on-line but verifies whether coins have been double-spent when the payee deposits it in to his/her account.

A. Bitcoin

Bitcoin is a decentralised digital cash system which works on a peer to peer network. The system was first proposed and developed by Satoshi Nakamoto (pseudonym) in October 2008. Bitcoin system is widely used to make anonymous payments over the Internet. Every payment that occurs in the Bitcoin network is broadcast to all network nodes. The process of Bitcoin peers authorising transactions and creating new Bitcoins is called Bitcoin mining. A user can generate a unique Bitcoin address also known as a Bitcoin public key by using an Elliptic Curve Digital Signature Algorithm (ECDSA) key [6], [16]. A Bitcoin transaction is the process of transferring a Bitcoin by digitally signing a hash of a previous transaction together with

978-1-4799-6563-2/14 $31.00 © 2014 IEEE
DOI 10.1109/ICEBE.2014.20

the next owners public Bitcoin address and adding this record to the end of the coin. This chain of signature ownership links past transactions to the present transactions. The creation of blocks and verification of Bitcoins by peers in the network is made a fair and non-trivial task by introducing a *Proof of Work* method. With this concept, peers needs to generate hashes until a hash value is generated with a certain number of zero bits at the beginning of the hash. The work needed increases exponentially as the required number of bits in the beginning increases. As a result, a constructed block cannot be easily changed [16]. The security of Bitcoin transactions rely on the correctness of the Block Chain [16], [6]. Bitcoin transactions are anonymous and irreversible. A consumer would not be able to request a refund or prove to any party that the product was not delivered. The proposal in this paper is designed to address this concern.

III. Fair-exchange Protocol Schemes

Fairness can be categorised as Weak and Strong fairness. Weak fairness is when two parties engage in an electronic transaction, after the protocol-run, the honest party can prove to an external judge that he/she followed the protocol even though the dishonest party did not [17]. Strong fairness or true fairness ensures that the protocol itself tries to resolve disputes and misbehaving of parties without having to reach an external judge. These protocols make sure that a honest party engaged in a transaction does not get penalised when a dishonest party misbehaves [1]. Protocols that are built to achieve fairness in e-commerce transactions are called *Fair-exchange Protocols.* A combined solution that would realise fairness as well as anonymity is called an *Anonymous Fair-exchange Protocol.* Only few Anonymous Fair-exchange Protocols for exchanging electronic content have been proposed in academic literature [13], [21], [22], [20]. Fair-exchange protocols can be divided in to two main categories considering the involvement of a trusted third party (TTP) or not. Two party protocols do not rely on a TTP to achieve fairness. Examples include [7], [4]. However, these protocols lack simultaneity of exchange where parties could misbehave for their own advantage during transactions. TTP based fair-exchange protocols can be classified into three categories. **In-line**: Involves the TTP to collect exchanged items, check their accuracy and finally forward them to the intended parties. Proposals include [3], [11]. The involvement of the TTP, provides strong fairness. However, the TTP is required to be available any-time to manage and maintain large databases of communicated messages which can be considered as a major bottleneck. **On-line**: Similar to the above but the TTP does not involve in every transmitted message. However, TTP still engages in the protocol run to guarantee fairness by validating, storing and generating transmitted messages. Examples include [12], [23], [22]. **Off-line**: Lets the transacting parties exchange products without the intervention of a TTP unless one of the parties misbehaves, prematurely aborts or a communication failure happens. These protocols are also called *"Optimistic Protocols"*. Examples include [8], [2], [13]. The proposed protocol in this paper keeps the involvement of a TTP to a minimum by using an Off-line TTP.

IV. Proposed Protocol

The objectives of the proposed protocol are listed below. Mentioned in Table I is the notation used in the proposed protocol. Illustrated in Figure 1 is a diagram of the proposed protocol's message flow between the protocol entities.

★ The protocol should achieve strong fair-exchange while preserving anonymity of the consumer and the merchant.
★ TTP should not be able to see the exchanged product or store a copy of it.
★ Guarantee security properties such as; confidentiality, integrity, message freshness and non-repudiation.
★ Keep the involvement of the TTP to a minimum.
★ Disputes should be resolved within the protocol.
★ The protocol should support similar digital cash systems.

TABLE I. Notation used in the proposed protocols

$C/M/TTP/PV$	:	Consumer / Merchant / Trusted Third Party / Product Verifier.
$BP2P$	:	The Bitcoin Peer-to-Peer Network.
T_i	:	Purchase/delivery transaction of product m by C and M.
$Pseudo\text{-}ID\text{-}M$	:	Unique Pseudonym-Identity of M registered with the PV.
$Pseudo\text{-}ID\text{-}iX$	:	Unique Pseudonym-Identity of X registered with the TTP, only used during T_i.
K_1	:	Public encryption key of the public/private key pair escrowed with TTP, later used by PV to encrypt m.
K_1^{-1}	:	Private decryption key of the public/private key pair escrowed with TTP. The key pair is generated by M.
$eK_1\{Z\}$	:	Encryption of data string Z using a public algorithm with K_1.
P_X	:	Public Encryption Key of entity X.
$eP_X\{Z\}$	:	Encryption of data string Z using a public algorithm with the public encryption key P_X of entity X.
S_X	:	Private Signature Key of entity X.
$sS_X\,[Z]$	:	Digital signature outcome (without message recovery) from applying the private signature transformation on data string Z using the private signature key S_X of entity X.
V_X	:	Public Signature Verification Key of entity X.
P_{wM}	:	Public Encryption Key of M advertised online with m.
P_{iX}	:	Public Encryption Key of entity X used only during T_i.
$eP_{iX}\{Z\}$	:	Encryption of data string Z using a public algorithm with the public encryption key P_{iX} of X used only during T_i.
S_{iX}	:	Private Signature Key of entity X used only during T_i.
$sS_{iX}\,[Z]$	:	Digital signature outcome (without message recovery) from applying the private signature transformation on data string Z using S_{iX} of X only during T_i.
V_{iX}	:	Public Signature Verification Key of X only during T_i.
$V_{iX_{cert}}$	:	Public Signature Verification certificate issued by the TTP. It includes V_{iX} corresponding to the $Pseudo\text{-}ID\text{-}iX$ of entity X used only during T_i.
BP_X	:	Bitcoin Public Key of entity X (X's Bitcoin address).
BS_X	:	Bitcoin Private Key of entity X (X's Signature key).
$sBS_X\{Z\}$	:	Digital Signature outcome (without message recovery) from applying the private signature transformation on data string Z using BS_X of X.
$T\text{-}info$	:	Other information relevant to a particular Bitcoin transaction.
T_X	:	Bitcoin transaction from C to M, in the formation of a hash. $T_X = h(T_{(X-1)}\|\|BP_C\|\|T\text{-}info)$.
T_{X-1}	:	Previous Bitcoin transaction that has occurred in the past but directly linked to T_X in the formation of a hash.
$Encrypt_{cert}$	:	Encryption certificate issued by the PV. It includes a hash of the encrypted product which has been encrypted by the PV using the key indicated in the certificate.
TTP_{commit}	:	Commitment certificate issued by the TTP indicating involvement in the exchange.
$TTP\text{-}Pool$	:	A pool of different TTPs. C & M agrees between one TTP from this list to be involved in T_i.
PV_{cert}	:	Product verification certificate issued by the PV.
$t_{-payment}$	:	Predefined time-out for M to send the decryption key, includes time needed for Bitcoin transaction processing.
t	:	Predefined time-out period agreed by involved parties. If a response is not received within the time-out the sending party will resend once more, in case a no reply, the sending party aborts the protocol or involve the TTP if necessary.
$t_{-resolve}$	:	Time-out given to M by TTP to respond with the requested key before sending the escrowed product decryption key to C.
$A\|\|B$	:	Concatenation of A and B in that order.
$h(Z)$	:	Hash of data string Z.
N_{1X} / N_{2X}	:	First & second nonce issued by entity X.
$X \rightarrow Y : Z$	:	Entity X sends message Z to entity Y.

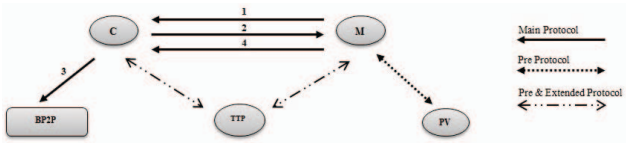

Fig. 1. Proposed protocol message flow.

The protocol can be broken down into three stages. The first stage is the *"Pre Protocol"* where product registration, product/price negotiation, TTP selection and product encryption happens. The second stage is the *"Main Protocol"* where product delivery, Bitcoin payment and decryption key delivery happens. If things go according to plan the protocol completes in the above two stages, but if a transacting party misbehaves, prematurely aborts or a communication failure happens the third stage *"Extended Protocol"* is executed with the involvement of the TTP. The assumptions mentioned in Table II have been taken in to account for a successful run of the protocol.

TABLE II. ASSUMPTIONS IN THE PROPOSED PROTOCOLS

$A1$	:	M registers with PV by giving a $Pseudo\text{-}ID\text{-}M$, P_M and V_M which are only used in communication between M and PV. The PV certifies M's public verification key V_M and keep a record of it to verify M's messages signed using S_M in future communication.
$A2$	:	Both C and M register with TTP by giving a per-transaction pseudonym-identity $Pseudo\text{-}ID\text{-}iC$ and $Pseudo\text{-}ID\text{-}iM$. The TTP makes sure that each Pseudo-ID is unique and has not been registered before. It should not be possible for TTP, C, M or external parties to deduce the real identity of C and M by examining the Pseudo-ID.
$A3$	:	C's and M's public verification keys V_{iC} and V_{iM} are certified by the TTP to their pseudonym-identities $Pseudo\text{-}ID\text{-}iC$ and $Pseudo\text{-}ID\text{-}iM$ respectively. The public certificates $V_{iC_{cert}}$ & $V_{iM_{cert}}$ are issued to each owner by TTP and can later be used to verify each other's digital signatures.
$A4$	:	C and M have access to a Bitcoin wallet. M generates a one-time Bitcoin address to be presented to C to receive payments and only if needed, C generates a one-time Bitcoin address to receive any change back form the transaction also known as a change-address.
$A5$	:	C and M, in addition to pseudonyms, maintain anonymity by setting up Anonymity Channels (uses cryptographic processes to change message origin details and prevent eavesdropping) for communication.
$A6$	:	All cryptographic keys are checked for validity before use, standardised public key algorithm (e.g. RSA) is used for encryption, data is padded according to recommended best practice before encryption, hashes are generated using standardised secure hash functions (e.g. SHA) and messages are signed using a standardised digital signature algorithm (e.g. DSA).

A. Pre Protocol Stage

The Pre Protocol messages are listed in Table III and described in detail below.

Firstly, M advertises the unique $product\text{-}ID$ generated by M, $product\text{-}price$, $product\text{-}description$ and a public key P_{wM} online. The website M uses to advertise these details does not necessarily have to be his/her own site but could also be a third party listing service. M also advertises a $TTP\text{-}Pool$ which is a list of TTPs that could be involved in the transaction. A consumer wishing to purchase a product, selects a TTP from $TTP\text{-}Pool$ and registers with it according to assumption *A2*. This gives C control over the selection of a TTP then having to rely on a single TTP proposed by M.

Message a: After registration with TTP, C creates a concatenation which includes; $product\text{-}ID$, a purchase order, a digital signature on the hash of $Order$ using S_{iC}, P_{iC} only used in T_i, fresh nonce generated by C and the public signature verification certificate issued by TTP. The concatenation is

TABLE III. PRE PROTOCOL STAGE

a.	$C \rightarrow M$	:	$eP_{wM}\{product\text{-}ID \vert\vert Order \vert\vert sS_{iC}[h(Order)] \vert\vert P_{iC} \vert\vert N_{1C} \vert\vert V_{iC_{cert}}\} \vert\vert sS_{iC}[h(encryption)]$ $Order$=$Pseudo\text{-}ID\text{-}iC \vert\vert TTP \vert\vert payment\text{-}method \vert\vert product\text{-}price$
b.	$M \rightarrow TTP$	:	$eP_{TTP}\{Transaction\text{-}ID \vert\vert K_1 \vert\vert K_1^{-1} \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert N_{1M}\} \vert\vert sS_{iM}[h(encryption)]$
c.	$TTP \rightarrow M$	:	$eP_{iM}\{K_1 \vert\vert TTP \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert sS_{TTP}[h(K_1 \vert\vert TTP \vert\vert Pseudo\text{-}ID\text{-}iM)] \vert\vert Transaction\text{-}ID \vert\vert N_{1M} \vert\vert N_{1TTP}\} \vert\vert sS_{TTP}[h(encryption)]$
d.	$M \rightarrow PV$	:	$eP_{PV}\{pseudo\text{-}ID\text{-}M \vert\vert product\text{-}description \vert\vert m \vert\vert P_M \vert\vert product\text{-}ID \vert\vert N_{2M} \vert\vert K_1 \vert\vert TTP \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert sS_{TTP}[h(K_1 \vert\vert TTP \vert\vert Pseudo\text{-}ID\text{-}iM)]\} \vert\vert sS_M[h(encryption)]$
e.	$PV \rightarrow M$	:	$eP_M\{PV_{cert} \vert\vert Encrypt_{cert} \vert\vert N_{2M} \vert\vert N_{1PV}\} \vert\vert sS_{PV}[h(encryption)]$ PV_{cert}=$X_1 \vert\vert sS_{PV}[h(X_1)]$ X_1=$product\text{-}ID \vert\vert product\text{-}description \vert\vert eK_1\{m\}$ $Encrypt_{cert}$=$X_2 \vert\vert sS_{PV}[h(X_2)]$ $X_2 = h(eK_1\{m\}) \vert\vert K_1 \vert\vert TTP \vert\vert Pseudo\text{-}ID\text{-}iM$
f.	$M \rightarrow TTP$	:	$eP_{TTP}\{Transaction\text{-}ID \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert Pseudo\text{-}ID\text{-}iC \vert\vert Encrypt_{cert} \vert\vert N_{1TTP} \vert\vert N_{3M}\} \vert\vert sS_{iM}[h(encryption)]$
g.	$TTP \rightarrow M$	:	$eP_{iM}\{TTP_{commit} \vert\vert N_{3M} \vert\vert N_{2TTP}\} \vert\vert sS_{TTP}[h(encryption)]$ TTP_{commit}=$Y_1 \vert\vert sS_{TTP}[h(Y_1)]$ Y_1=$Transaction\text{-}ID \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert Pseudo\text{-}ID\text{-}iC \vert\vert h(eK_1\{m\})$

then encrypted using M's advertised public key. The hash of this encryption is signed by C using S_{iC} to create a digital signature, represented here as $sS_{iC}[h(encryption)]$. C appends the digital signature to the encryption and send both parts to M. For the rest of the protocol descriptions we use the same notation to represent digital signatures sent in each message. M could verify the digital signatures using $V_{iC_{cert}}$. The $Order$ includes; $Pseudo\text{-}ID\text{-}iC$ registered with TTP used only during T_i, TTP chosen and registered by C, $Payment\text{-}method$ to indicate which digital cash system to use and $product\text{-}price$.

Message b: M after receiving C's message, registers with the same TTP according to *A2*. M creates a concatenation which includes; a unique transaction ID generated by M for T_i, public encryption key to be escrowed with TTP, private decryption key to be escrowed with TTP, $Pseudo\text{-}ID\text{-}iM$ registered with TTP used only during T_i and a fresh nonce generated by M. The concatenation is encrypted using P_{TTP}. M signs the hash of the encryption using S_{iM} and appends it to the encryption before sending it to TTP.

Message c: TTP after receiving M's message, checks the compatibility of the public/private key pair to be escrowed. If satisfied, creates a concatenation which includes; K_1 , TTP, $Pseudo\text{-}ID\text{-}iM$ (we refer to these as "the three components"), TTP's digital signature on the hash of the three components, the transaction ID, M's nonce and a newly generated nonce by the TTP. The concatenation is then encrypted using P_{iM}. TTP signs the hash of the encryption using S_{TTP} and appends it to the encryption before sending it to M.

Message d: Upon receiving TTP's message and successful registration with PV according to *A1*, M now needs to get product m certified and encrypted by the PV using K_1 escrowed with TTP. M creates a concatenation which includes; $pseudo\text{-}ID\text{-}M$, $product\text{-}description$, m, P_M, $product\text{-}ID$,

newly generated nonce by M and the three components & TTP's digital signature on the hash of the three components. M encrypts the concatenation using P_{PV}. M signs the hash of the encryption using S_M only used with PV according to *A1* and appends it to the encryption before sending it to PV .

Message e: Upon receiving the message, PV checks whether the product matches its product-description. If it matches, PV encrypts m using K_1 and generates a product verifier certificate PV_{cert} which includes X_1 and a signed hash of X_1 using S_{PV}. X_1 consists of the product−ID, product−description and encrypted product. At the same time, PV also verifies TTP's digital signature on the three components received in the previous message. If satisfied, PV generates an *Encryption Certificate*. The $Encrypt_{cert}$ includes X_2 and a digital signature on the hash of X_2 using S_{PV}. X_2 consists of a hash of the encrypted product and the three components verified to have come from TTP. PV then creates a concatenation which includes; PV_{cert}, $Encrypt_{cert}$, M's nonce N_{2M} and a newly generated nonce N_{1PV}. The concatenation is then encrypted using P_M shared only with PV. PV signs the hash of the encryption using S_{PV} and appends the it to the encryption before sending it to M.

Message f: M now creates a concatenation which includes; the $Transaction\text{-}ID$, $Pseudo\text{-}ID\text{-}iM$, $Pseudo\text{-}ID\text{-}iC$, $Encrypt_{cert}$, N_{1TTP} and a newly created nonce by M. M encrypts the concatenation using P_{TTP} and appends a signed hash of the encryption using S_{iM} before sending it to TTP. Here with the $Encrypt_{cert}$, the TTP only receives a hash of the encryption but not the actual encrypted product.

Message g: Lastly, TTP verifies $Encrypt_{cert}$ which indicates that the product was encrypted using key K_1 escrowed with TTP. The TTP issues a commitment certificate called that TTP_{commit}. TTP creates a concatenation which includes; the TTP_{commit}, M's previous nonce and N_{2TTP}. TTP encrypts the concatenation using P_{iM} and appends a signed hash of the encryption using S_{TTP} before sending it to M. The TTP_{commit} includes Y_1 and a digital signature of TTP by signing the hash of Y_1 using S_{TTP}. Y_1 consists of the $Transaction\text{-}ID$, $Pseudo\text{-}ID\text{-}iM$, $Pseudo\text{-}ID\text{-}iC$ and a hash of the encrypted product.

B. Main Protocol Stage

Following completion of the pre protocol stage, M examines the TTP_{commit} received from the TTP and initiates the main protocol. Main protocol messages are listed in Table IV and described in detail below.

Message 1: M creates a concatenation which includes; the $product\text{-}ID$, a newly created $Invoice$, a digital signature by signing the hash of the $Invoice$ using S_{iM} to indicate that M agrees with the terms of the transaction, N_{1C}, newly generated nonce, PV_{cert}, TTP_{commit}, M's public signature verification certificate issued by TTP and a predefined time-out. M encrypts the concatenation using P_{iC} and appends a digital signature created by signing the hash of the encryption using S_{iM}. The $Invoice$ consists of the $Transaction\text{-}ID$, $product\text{-}price$, $payment\text{-}method$, $Pseudo\text{-}ID\text{-}iC$, $Pseudo\text{-}ID\text{-}iM$, TTP and M's one-time Bitcoin public key (address).

C after receiving M's message, decrypts the message and retrieves $V_{iM_{cert}}$ to verify M's digital signature. C verifies that

TABLE IV. MAIN PROTOCOL MESSAGES.

1.	$M \rightarrow C$	: $eP_{iC}\{product\text{-}ID\|\|Invoice\|\|sS_{iM}[h(Invoice)]$ $\|\|N_{1C}\|\|N_{4M}\|\|PV_{cert}\|\|TTP_{commit}\|\|V_{iM_{cert}}\|\|\mathfrak{t}\}$ $\|\|sS_{iM}[h(encryption)]$ $Invoice$=$\{Transaction\text{-}ID\|\|product\text{-}price$ $\|\|payment\text{-}method\|\|Pseudo\text{-}ID\text{-}iC\|\|Pseudo\text{-}ID\text{-}iM\|\|TTP\|\|BP_M\}$
2.	$C \rightarrow M$	: $eP_{iM}\{Invoice\|\|sS_{iM}[h(Invoice)]$ $\|\|sS_{iC}[h(sS_{iM}[h(Invoice)])]$ $\|\|N_{4M}\|\|N_{2C}\|\|h(eK_1\{m\})\|\|\mathfrak{t}_{-payment}\}$ $\|\|sS_{iC}[h(encryption)]$
3.	$C \rightarrow BP2P$	: $\{amount\|\|BP_C\|\|BP_M\|\|sBS_C[T_X\|\|BP_M]$ $\|\|T_X\|\|T\text{-}info\}$ $T_X = h(T_{(X-1)}\|\|BP_C\|\|T\text{-}info)$
4.	$M \rightarrow C$	: $eP_{iC}\{Invoice\|\|N_{2C}\|\|N_{5M}\|\|h(eK_1\{m\})$ $\|\|K_1^{-1}\|\|\mathfrak{t}\}$ $\|\|sS_{iM}[h(encryption)]$

PV_{cert} using PV's digital signature and TTP_{commit} using TTP's digital signature appended in the commitment certificate. The latter assures that TTP has confirmed involvement in the fair-exchange. C then carries out two main verification steps before proceeding further; **Firstly**, C obtains $product\text{-}ID$ and $product\text{-}description$ from PV_{cert} and compares them with the details of the product C is willing to purchase as advertised by M. If condition ① shown below is satisfied then C is certain that the encrypted product m and it's details as certified by the PV is the intended product that he/she is about to pay for. **Secondly**, C generates a hash of the encrypted product $eK_1\{m\}$ and compares it with the hash obtained from TTP_{commit}. If both hashes match, then condition ② shown below is satisfied. This confirms that the hash of the encrypted product $eK_1\{m\}$ matches the hash TTP has confirmed to have the corresponding decryption key for. This gives assurance to C that after making a payment, if M misbehaves, prematurely aborts or communication fails, the product decryption key can still be obtained by initiating the extended protocol. If and only if conditions ① & ② are satisfied, C proceeds to message 2 or else C aborts the protocol and informs both M & TTP the reasons.

$Advertised\ Details = Details\ found\ in\ PV_{cert}$ ——— ①
$Computed\ h(eK_1\{m\}) = h(eK_1\{m\})\ found\ in\ TTP_{commit}$ ——— ②

Message 2: C creates a concatenation which includes; $Invoice$, M's digital signature on the invoice, a digital signature by signing the hash of M's digital signature $sS_{iM}[h(Invoice)]$ using S_{iC} to indicate that C agrees with the terms of the transaction, N_{4M}, newly generated nonce by C, hash of the encrypted product and a predefined time-out $\mathfrak{t}_{-payment}$. The concatenation is then encrypted using P_{iM} and a signed hash of the encryption using S_{iC} is appended before sending it to M.

Message 3: Immediately after sending message 2, C makes a Bitcoin payment to M's Bitcoin address BP_M. T_X is a hash of the Bitcoin transaction from C to M. The hash includes the previous transaction hash linking to this transaction, Bitcoin public key of C and other transaction information relevant to this transaction. The message includes the transferring amount, C's one-time Bitcoin public key, M's one-time Bitcoin public key, digital signature created by signing the transaction hash T_X using C's Bitcoin private signature key BS_C , a transaction hash T_X and other information related to the transaction.

Upon receiving, random peers in the Bitcoin network start creating a new block with the transaction information as follows; A peer computes a new block which includes the hash $T_{(X+1)}$ from this transaction. In $BP2P$ every peer engages in computing blocks simultaneously. Due to this reason only the first valid block created is verified by other peers to be genuine and a new record is added to the Block Chain. The Bitcoin network at the same time checks whether the Bitcoins have been spent previously to detect double spending.

Message 4: M after receiving C's Bitcoin payment, now needs to send C the product decryption key K_1^{-1}. M creates a concatenation which includes; the $Invoice$, N_{2C}, a newly generated nonce, $h(eK_1\{m\})$, K_1^{-1} and a time-out. The concatenation is then encrypted using P_{iC}. M signs the hash of the encryption using S_{iM} and appends it to the encryption before sending. If the main protocol messages complete successfully then C decrypts the product and M updates $Transaction\text{-}ID$ as completed in his/her record.

C. Extended Protocol Stage

In the event of after C making a Bitcoin payment if M misbehaves by sending an incorrect decryption key, prematurely aborts or a communication failure happens then C initiates the extended protocol as listed in Table V and described below.

TABLE V. EXTENDED PROTOCOL MESSAGES.

I.	$C \rightarrow TTP$	: $eP_{TTP}\{BlockchainEvidence\|\|Invoice \|\|sS_{iM}[h(Invoice)]\|\|N_{3C}\|\|h(eK_1\{m\}) \|\|TTP_{commit}\|\|\mathfrak{t}\} \ \|\|sS_{iC}[h(encryption)]$
II.	$TTP \rightarrow M$	: $eP_{iM}\{BlockchainEvidence\|\|Invoice \|\|sS_{iM}[h(Invoice)]\|\|KeyRequest \|\|N_{3TTP}\|\|\mathfrak{t}_{-resolve}\} \ \|\|sS_{TTP}[h(encryption)]$
III.	$M \rightarrow TTP$	: $eP_{TTP}\{Invoice\|\|K_1^{-1}\|\|N_{3TTP}\|\|N_{6M}\|\|\mathfrak{t}\} \|\|sS_{iM}[h(encryption)]$
IV.	$TTP \rightarrow C$	: $eP_{iC}\{Invoice\|\|K_1^{-1}\|\|N_{3C}\|\|N_{4TTP}\|\|\mathfrak{t}\} \|\|sS_{TTP}[h(encryption)]$

Message I: C creates a concatenation which includes; the blockchain evidence showing the Bitcoin transfer from the C to M's Bitcoin address, $Invoice$, M's digital signature on the hash of $Invoice$ received in message 1, a newly generated nonce by C, hash of the encrypted product generated by C, TTP_{commit} and a predefined time-out. C encrypts the message using P_{TTP} and appends a digital signature using S_{iC}.

Message II: TTP after receiving the previous message, examines the evidence. The TTP_{commit} confirms that TTP is involved in the transaction and $sS_{iM}[h(Invoice)]$ confirms that M has agreed to T_i. The $BlockchainEvidence$ is verified by TTP by looking in to the Bitcoin BlockChain and it confirms the Bitcoin payment from C to M. TTP in response, creates a concatenation which includes; the $BlockchainEvidence$, $Invoice$, M's digital signature on the hash of the $Invoice$, a $KeyRequest$ from the TTP requesting for K_1^{-1}, a newly created nonce by the TTP and a predefined time-out $\mathfrak{t}_{-resolve}$. TTP encrypts the concatenation using P_{iM} and appends a signed hash of the encryption using S_{TTP} before sending it to M.

Message III: If M has not disappeared after receiving C's payment, misbehaved or deliberately refusing to communicate, then once TTP's key request is received, M checks the status of the $Transaction\text{-}ID$. If it has not completed, M creates a concatenation which includes; $Invoice$, the product decryption key, N_{3TTP}, a newly generated nonce by M and a predefined time-out. M encrypts the concatenation using P_{TTP} and appends a signed hash of the encryption using S_{iM} before sending it to TTP.

Message IV: If M responds to TTP's key request within $\mathfrak{t}_{-resolve}$, TTP proceeds to message IV as normal but if M's response is not received then TTP retrieves the product decryption key escrowed with itself and create message IV. In either scenario, TTP creates a concatenation which includes; $Invoice$, K_1^{-1}, N_{3C}, a newly generated nonce and a time-out. TTP encrypts the concatenation using P_{iC} and appends a signed hash of the encryption using S_{TTP}. C after receiving this message can successfully decrypt the product.

V. SECURITY ANALYSIS

In this section, the proposed protocol is reviewed to see whether it achieves the objectives outlined in Section IV.

★ **Strong Fair-Exchange**: It should be noted that if a party aborts the protocol before C makes the Bitcoin payment in *Message 3*, fair-exchange is not affected as neither party gains an advantage from the other. This means that M will not be made a Bitcoin payment and C will not be able to decrypt the product. If both C and M followed the protocol without misbehaving or prematurely aborting, the protocol completes without an extended protocol stage. The following scenarios are also taken into account;

M sends a wrong encrypted product: M gains no advantage by this act as C only proceeds to make a payment once conditions ① & ② are satisfied.
M sends a wrong product decryption key: In this scenario C initiates the extended protocol by sending all gathered evidences to TTP as shown in *Message I* in the dispute resolution. At the end of the extended protocol TTP forwards C the correct decryption key.
M after receiving payment demands more payment: In this scenario C initiates the extended protocol and TTP by looking in to evidence C has provided in *Message I* would determine that M agreed to the terms of the transaction and the price by digitally signing the $Invoice$.
M disappears after receiving payment or aborts: The extended protocol is initiated by C.
C pays *M* less than the agreed amount: M only sends the decryption key if full payment is received. Due to this C doesn't gain any advantage by making partial payments.
C pays *M* less and initiates the extended protocol: If C claims deceivingly that the amount paid is what he/she agreed, then upon enquiry by TTP in *Message III*, M could send signed purchase order $sS_{iC}[h(Order)]$ in *Message c*, and if received $sS_{iC}[h(sS_{iM}[h(Invoice)])]$ in *Message 2*, which both includes C's digital signature agreeing to the terms of the transaction.
C collude with *TTP*: Since C chooses the TTP, C is more likely to collude with TTP than M. This may disadvantage M as TTP could send C the escrowed decryption key before C's Bitcoin payment to M. However, anonymity of M cannot be breached this way as M never reveals the real identity.
M collude with *TTP*:This may disadvantage C as TTP could send C an incorrect decryption key in the extended

stage after C's Bitcoin payment to M. However, anonymity of C is not breached.

C and *M* collude together: As both parties are dishonest there is no fair-exchange to be achieved.

Considering the above analysis it is evident that the protocol guarantees that if one party misbehaves the other party doesn't incur any loss or have to resolve any disputes with the manual intervention of an external judge after the protocol. Instead, disputes can be resolved within the protocol itself. All these properties constitute strong fair-exchange.

★ **Anonymity**: Both C and M never reveal their real identities or personal details at any stage of the protocol. Instead, they use per-transaction pseudo-IDs and public/private key pairs. M interacts with C and TTP using $Pseudo\text{-}ID\text{-}iM$ only used in T_i and with PV using $pseudo\text{-}ID\text{-}M$ for product registrations. C interacts with M and TTP using $pseudo\text{-}ID\text{-}iC$ and does not interact with PV. Due to these aspects anonymity of C and M is not only guaranteed between each other but also to PV and TTP. A party trying to collude with PV or TTP to find the real identity of the other, would not gain any benefit other than what they already know. e.g. M colluding with TTP to identify C. In addition to pseudonyms both C and M set-up anonymity channels for communication between each other and other parties as in *A5*. It is common practise while making Bitcoin payments to use an anonymiser such as the *TOR Browser*. C makes the Bitcoin payment to a one-time Bitcoin address generated by M. Similarly C generates a one-time Bitcoin address to receive Bitcoins as change only if there are any. TTP does not get to know about the Bitcoin transaction and addresses unless the extended protocol is executed.

By looking at one's Bitcoin address, the true identity of the user cannot be revealed. However, due to the necessity of having to broadcast all transactions publicly prevents the anonymity of Bitcoin payment transactions. This has become a drawback and with the advancements in computational power and access to *Data Analysis* capabilities, it may be possible to link Bitcoin transactions to real user identities [18], [14]. Therefore, in Section VI, we add an extension to our protocol to support Zerocoin/Zerocash system proposed in [15], [5] to provide improved privacy grantees and full anonymity.

★ **Privacy of exchanged product**: TTP involved does not get to see m or keep a copy of it. TTP does not get to know any information related to product m until the extended protocol is initiated. Even then the details included in $Invoice$ are not sufficient for TTP to find out what was exchanged.

★ **Security properties**: The protocol messages are encrypted to provide confidentiality and a nonce confirms message freshness. Registered pseudonym-IDs and digital signatures provide non-repudiation. The digital signatures appended provides message integrity. The predefined time-outs provide timeliness and lets a sending party to resend a message once more if a response is not received, to complete the protocol by aborting or to complete the protocol by resolving with the TTP without letting the protocol go in to an infinite loop.

A. Other Properties

★ **Minimum involvement of *TTP***: The protocol keeps the involvement of the TTP to a minimum by using an off-line TTP. The TTP is not required to intervene in the protocol unless a transacting party misbehaves, prematurely aborts or a communication failure happens, which makes the protocol optimistic.

★ **Dispute resolution**: Disputes are resolved automatically within the protocol run without manual intervention of an external judge.

★ **Other digital cash support**: Even though more emphasis was given to Bitcoin as the payment method for the protocol, it also can be adapted to support other digital cash systems that are designed using a public ledger/block chain. A generalised example of how the protocol can be adapted to support other digital cash systems similar to Bitcoin designed with the concept of providing anonymous payments using pseudonymous addresses and publishing every transaction in a public ledger is explained here. C informs M the currency that he/she would like to use in the $payment\text{-}method$ in *Message a*. M later sends an address relevant to the selected payment method in *Message 1*. C makes a payment to M using the chosen payment system. If things go according to plan and M sends the decryption key then the protocol completes without further changes. However to support the extended protocol stage, C could present evidence from the relevant public ledger of the digital currency system in *Message I*. This evidence may include the publicly available transaction record of the payment made from one-time address of the payer to the one-time address of the payee. This provides enough evidence for the TTP to continue with the rest of the protocol without further changes. In Section VI a more detailed explanation is given while extending our protocol to support Zerocoin/Zerocash which works differently to the generalised example due to increased anonymity as well as privacy of transaction details published in the block chain.

VI. Extension to support Zerocoin/Zerocash

To make cross-referring easy for the reader, we slightly divert from our protocol notation to match the notation in corresponding papers [5], [15].

TABLE VI. Notation used for Zerocoin/Zerocash extension

$ZP2P$	:	Zerocash System integrated into Bitcoin P2P Network.
PRF	:	Pseudorandom function.
$COMM$	:	Statistically-hiding non-intractive commitment scheme.
$_{zk\text{-}SNARK}$	:	zero-knowledge Succinct Non-interactive ARguments of Knowledge.
rt	:	A root of Merkle tree at a given time.
a_{pk_X}	:	Address Public Key of entity X (X's Zerocoin address).
a_{sk_X}	:	Address Secret Key of entity X.
z_x	:	A Zerocoin that is owned by entity X.
z_x^{old}	:	A Zerocoin that is owned by entity X and is used to pour it's value to new coins.
v_x	:	The value of entity X's Zerocoin.
v_{pub}	:	A non-negative public output value that can be used to pay a target similar to a Bitcoin address as specified in a transaction string $info$.
ρ_x	:	A secret value that determines sn_x of entity X's coin.
sn_x	:	A serial number derived as $sn_x = PRF^{sn}_{a_{sk_X}}(\rho_x)$.
cm_x	:	A coin commitment of entity X's coin (a string that appears in the public ledger) constructed as $k_x = COMM_{r_x}(a_{pk_X}\|\|\rho_x)$ and $cm_x = COMM_{s_x}(v_x\|\|k_x)$ where r_x & s_x are random.
tx_{Mint}	:	A mint transaction records; when a new coin z with commitment cm and value v has been minted.
π_{POUR} :	:	A $zk\text{-}SNARK$ proof that states; "Given rt, old sn_x, new commitments cm_x and cm_y, I know coins z^{old}, new coins z_x, z_y and old address secret key a_{sk_X}".

tx_{Pour}	:	A pour transaction is used to spend, split, merge or transfer ownership of anonymous coins to others. Pour records the pouring of a old coin/two coins (z_x^{old}) with their corresponding serial numbers (sn_x^{old}) into two new coins (z_x, z_y) with their commitments (cm_x, cm_y) in the public ledger. It also records rt, v_{pub}, $info$ and π_{POUR}.

The modified protocol messages, in order to support our extension to Zerocash system are listed in Table VII and described in detail below.

TABLE VII. PROTOCOL EXTENSION TO ZEROCOIN/ZEROCASH.

f. $M \rightarrow TTP$	:	$eP_{TTP}\{Transaction\text{-}ID \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert Pseudo\text{-}ID\text{-}iC \vert\vert Encrypt_{cert} \vert\vert N_{1TTP} \vert\vert N_{3M} \vert\vert \mathbf{cm_m}\}\ \vert\vert sS_{iM}[h(encryption)]$
g. $TTP \rightarrow M$	:	TTP_{commit}=$Y_1 \vert\vert sS_{TTP}[h(Y_1)]$ Y_1=$Transaction\text{-}ID \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert Pseudo\text{-}ID\text{-}iC \vert\vert h(eK_1\{m\}) \vert\vert \mathbf{cm_m}$
1. $M \rightarrow C$	:	$Invoice$=$\{Transaction\text{-}ID \vert\vert product\text{-}price \vert\vert payment\text{-}method \vert\vert Pseudo\text{-}ID\text{-}iC \vert\vert Pseudo\text{-}ID\text{-}iM \vert\vert TTP \vert\vert \mathbf{a_{pk_m}} \vert\vert \mathbf{v_m} \vert\vert \rho_m \vert\vert \mathbf{r_m} \vert\vert \mathbf{s_m}\}$
3. $C \rightarrow ZP2P$	:	$\mathbf{z_m} = (\mathbf{a_{pk_m}} \vert\vert \mathbf{v_m} \vert\vert \rho_m \vert\vert \mathbf{r_m} \vert\vert \mathbf{s_m} \vert\vert \mathbf{cm_m})$ $\mathbf{z_c} = (\mathbf{a_{pk_c}} \vert\vert \mathbf{v_c} \vert\vert \rho_c \vert\vert \mathbf{r_c} \vert\vert \mathbf{s_c} \vert\vert \mathbf{cm_c})$ $\mathbf{tx_{Pour}} = (\mathbf{rt} \vert\vert \mathbf{sn^{old}} \vert\vert \mathbf{cm_m} \vert\vert \mathbf{cm_c} \vert\vert \pi_{\mathbf{POUR}} \vert\vert \mathbf{v_{pub}} \vert\vert \mathbf{info})$
I. $C \rightarrow TTP$	:	$eP_{TTP}\{\mathbf{cm_m} \vert\vert \mathbf{Invoice} \vert\vert sS_{iM}[h(Invoice)] \vert\vert N_{3C} \vert\vert h(eK_1\{m\}) \vert\vert TTP_{commit} \vert\vert \{t\}\ \vert\vert sS_{iC}[h(encryption)]$
II. $TTP \rightarrow M$	:	$eP_{iM}\{\mathbf{cm_m} \vert\vert Invoice \vert\vert sS_{iM}[h(Invoice)] \vert\vert KeyRequest \vert\vert N_{3TTP} \vert\vert \{t_{-resolve}\}\ \vert\vert sS_{TTP}[h(encryption)]$

Message f: In order for C to make a Zerocoin payment, M generates a new address key pair (a_{pk_m}, a_{pk_m}) and a secret value ρ_m. M now construct a coin commitment cm_m as k_m = $COMM_{r_m}(a_{pk_m}||\rho_m)$ and cm_m = $COMM_{s_m}(v_m||k_m)$ where r_m & s_m are random. M now appends cm_m to *Messages* f for TTP's record.

Message g: After receiving cm_m from the previous message, TTP keeps a record of cm_m and includes it as part of the commitment certificate TTP_{commit}. The purpose of this is that when C receives TTP_{commit} in *Messages* 1, it gives assurance for C that TTP is aware of the corresponding commitment of M's coin that is due to appear in the public ledger. C can also check the validity of cm_m by reconstructing it using secret values $a_{pk_m}, v_m, \rho_m, r_m, s_m$ received in *Messages* 1.

Message 1: M makes a slight change to the $Invoice$ to include the address public key a_{pk_m} instead of the Bitcoin address used in the previous protocol. M also includes secret values v_m, ρ_m, r_m, s_m to provide required information for c to make an anonymous payment to M's address. We let M generate these details instead of C to; 1) keep the protocol simple by not requiring a *key-private encryption scheme* to download these secret values in encrypted format from the public ledger as specified in the paper [5]. The main reason for this is that we have already established a secure channel in our protocol. 2) for M to generate the coin commitment and get it added to TTP_{commit} before C makes a payment.

Message 3: Immediately after sending *Messages* 2, C prepares to make a Zerocoin payment using the Zerocash system. Assuming C is a Bitcoin user, he/she now needs to deposit a Bitcoin with a backing escrow pool in the Zerocash system in order to mint a new Zerocoin (this step of minting a coin can be skipped if C already holds zerocoins). Due to the reason of C using this newly minted coin to make new coins, we add the notation old to it's parameters. Firstly, C generates a new address key pair ($a_{pk_c}^{old}, a_{pk_c}^{old}$) and a secret value ρ_c^{old} which determines coin z_c^{old}'s serial number sn_c^{old} = $PRF^{sn}_{a_{sk_c}^{old}}(\rho_c^{old})$. It is assumed that these serial numbers are collision resistant. C now generates cm_c^{old} as; k_c^{old} = $COMM_{r_c^{old}}(a_{pk_c}^{old}||\rho_c^{old})$ and cm_c^{old} = $COMM_{s_c^{old}}(v_c^{old}||k_c^{old})$ where r_c^{old} & s_c^{old} are random. The minting outputs a new coin and a mint transaction;

$$z_c^{old} = (a_{pk_c}^{old}||v_c^{old}||\rho_c^{old}||r_c^{old}||s_c^{old}||cm_c^{old})$$

$$tx_{Mint} = (v_c^{old}||k_c^{old}||s_c^{old}||cm_c^{old})$$

To spend the newly created coin, C carries out a *pour* operation which takes z_c^{old} as input coin and pours it's value into two fresh coins; z_m & z_c. C uses z_m to make M's payment and z_c to pay any *change back* from the transaction to him/her-self. To create these two coins C firstly, generates commitment cm_m for M's coin using the secret values received in *Messages* 1, such that; k_m = $COMM_{r_m}(a_{pk_m}||\rho_m)$ and cm_m = $COMM_{s_m}(v_m||k_m)$. C also at this point checks whether the constructed commitment matches the one found in TTP_{commit}. C now generates the commitment for his/her own new coin such that; k_c = $COMM_{r_c}(a_{pk_c}||\rho_c)$ and cm_c = $COMM_{s_c}(v_c||k_c)$ where r_c & s_c are random. Following this, C produces a π_{POUR} proof according to [5] and the serial number sn_c^{old} = $PRF^{sn}_{a_{sk_c}^{old}}(\rho_c^{old})$. The *pour* operation outputs two new coins and a tx_{Pour} that is appended to the public ledger.

$$z_m = (a_{pk_m}||v_m||\rho_m||r_m||s_m||cm_m)$$

$$z_c = (a_{pk_c}||v_c||\rho_c||r_c||s_c||cm_c)$$

$$tx_{Pour} = (rt||sn^{old}||cm_m||cm_c||\pi_{POUR}||v_{pub}||info)$$

M who is expecting a Zerocoin payment from C can now start using the received coin without having to scan the entire public ledger using the *Receive* algorithm beforehand as specified in [5]. This is due to M knowing cm_m and the private values that was used to generate it before receiving the payment. If M wants, the value of the received coin can be poured in to a new coin owned by M using the *pour* operation soon after receiving, making it's parameters such as the coin commitment only known to M or the value can be transferred to another owner as C did.

Messages I: If things go according to plan M sends the decryption key in *Messages* 4 and the protocol completes without going to an extended stage. However, after C making a payment, if M misbehaves by sending an incorrect decryption key, prematurely aborts or a communication failure happens then C appends cm_m and the new $Invoice$ instead of $BlockchainEvidence$ in *Messages I* to initiate the extended protocol. Note that TTP does not see $Invoice$ unless the protocol goes through to a extended phase.

Messages II: TTP after receiving C's message checks whether cm_m matches the one in TTP_{commit} and queries whether cm_m has appeared in the public ledger. TTP may use the secret values (v_m, ρ_m, r_m, s_m) found in the $Invoice$ for this task. If it has, this gives assurance to TTP that a payment was made to M's Zerocoin address corresponding

to the coin's commitment cm_m. If satisfied, TTP sends the $KeyRequest$ message to M which also includes cm_m. In either scenario, whether M forwards the product decryption key or not within the predefined time-out, TTP retrieves the escrowed decryption key and forwards it to C.

A. Security & Anonymity

Even though the previous protocol achieves anonymity using Bitcoin as the payment method, recent work has shown that it may be possible to link Bitcoin transactions to real identities [18], [14]. Our extension to support Zerocash addresses this issue while providing improved transaction privacy & anonymity for users. The protocol objectives as discussed in Section V are still achieved even though a separate analysis is not mentioned here due to space limitation. When C makes a payment to M using our extension the corresponding transaction record is not publicly available in the ledger. It should also be noted that despite the fact that C & TTP gets to know M's secret values (v_m, ρ_m, r_m, s_m) and commitment cm_m, coin z_m cannot be spend by either of these two parties as address secret key a_{sk_m} is only known by M. In addition when M spends z_m, it still cannot be traced as the output serial number $sn_m = PRF^{sn}_{a_{sk_m}}(\rho_m)$ is not revealed at any stage of our protocol.

VII. Conclusion & Future Work

The paper identified that genuine Bitcoin users are reluctant to make Bitcoin payments in e-commerce transactions due to transaction irreversibility. The proposed protocol achieves strong fair-exchange while preserving anonymity of the transacting parties. TTP agreed between C & M does not get to see the exchanged product or store a copy of it. The involvement of TTP is kept to a minimum and disputes are resolved within the protocol. The protocol can also be adapted to use other digital cash systems with public ledgers. We outline a drawback in Bitcoin that raises anonymity concerns and we add an extension to support Zerocash which addresses this issue while providing improved transaction privacy & anonymity for users. Future work could be outlined as; publishing formal analysis of the protocol that we are currently carrying out using Casper-FDR, making improvements in order to support exchange of physical products, further reduce the involvement of a significant TTP by using distributed TTPs.

References

[1] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, ser. CCS '97. New York, NY, USA: ACM, 1997, pp. 7–17. [Online]. Available: http://doi.acm.org/10.1145/266420.266426

[2] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange," in *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, 1998, pp. 86–99.

[3] A. Bahreman and D. Tygar, "Certified electronic mail," in *Network and Distributed Systems Security Conference*, February 1994, pp. 3–19.

[4] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest, "A fair protocol for signing contracts," *IEEE Transactions on Information Theory*, vol. 36, no. 1, pp. 40–46, 1990.

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on. IEEE*, 2014.

[6] Bitcoin.org. (2014) Bitcoin wiki. https://en.bitcoin.it/wiki/Main_Page. Bitcoin.org. [Online]. Available: https://en.bitcoin.it/wiki/Main_Page

[7] M. Blum, "How to exchange (secret) keys," *ACM Trans. Comput. Syst.*, vol. 1, no. 2, pp. 175–193, May 1983. [Online]. Available: http://doi.acm.org/10.1145/357360.357368

[8] H. Burk and A. Pfitzmann, "Digital payment systems enabling security and unobservability," *Comput. Secur.*, vol. 8, no. 5, pp. 399–416, Aug. 1989. [Online]. Available: http://dx.doi.org/10.1016/0167-4048(89)90022-9

[9] J. Camenisch, J.-M. Piveteau, and M. Stadler, "An efficient fair payment system," in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, ser. CCS '96. New York, NY, USA: ACM, 1996, pp. 88–94. [Online]. Available: http://doi.acm.org/10.1145/238168.238193

[10] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, Oct. 1985. [Online]. Available: http://doi.acm.org/10.1145/4372.4373

[11] T. Coffey and P. Saidha, "Non-repudiation with mandatory proof of receipt," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 1, pp. 6–17, Jan. 1996. [Online]. Available: http://doi.acm.org/10.1145/232335.232338

[12] B. Cox, J. D. Tygar, and M. Sirbu, "Netbill security and transaction protocol," in *Proceedings of the 1st Conference on USENIX Workshop on Electronic Commerce - Volume 1*, ser. WOEC'95. Berkeley, CA, USA: USENIX Association, 1995, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267185.1267191

[13] R. Indrakshi and R. Indrajit, "An optimistic fair exchange e-commerce protocol with automated dispute resolution," in *Electronic Commerce and Web Technologies*, ser. Lecture Notes in Computer Science, K. Bauknecht, S. Madria, and G. Pernul, Eds. Springer Berlin Heidelberg, 2000, vol. 1875, pp. 84–93. [Online]. Available: http://dx.doi.org/10.1007/3-540-44463-7_8

[14] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 127–140.

[15] I. Miers, C. Garman, M. Green, and A. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Security and Privacy (SP), 2013 IEEE Symposium on*, 2013, pp. 397–411.

[16] S. Nakamoto, "Bitcoin: A peer-to-peer e-cash system," *Bitcoin.org*, 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[17] I. Ray and I. Ray, "An anonymous fair exchange e-commerce protocol," in *In Proceedings of the International Workshop on Internet Computing and ECommerce*. IEEE Computer Society, 2001, pp. 172–179.

[18] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and Privacy in Social Networks*, Y. Altshuler, Y. Elovici, A. B. Cremers, N. Aharony, and A. Pentland, Eds. Springer New York, 2013, pp. 197–223. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-4139-7_10

[19] WorldPay, "Optimising your alternative payments: Global view," Whitepaper, 2010.

[20] N. Zhang, Q. Shi, and M. Merabti, "An efficient protocol for anonymous and fair document exchange," *Computer Networks*, vol. 41, no. 1, pp. 19 – 28, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128602003237

[21] Q. Zhang, K. Markantonakis, and K. Mayes, "A mutual authentication enabled fair-exchange and anonymous e-payment protocol," *E-Commerce Technology, IEEE International Conference on, and Enterprise Computing, E-Commerce, and E-Services, IEEE International Conference on*, vol. 0, p. 20, 2006.

[22] Q. Zhang, K. Mayes, and K. Markantonakis, *A user-centric m-payment solution*, 2005, p. 8.

[23] J. Zhou and D. Gollman, "A fair non-repudiation protocol," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, 1996, pp. 55–61.